MAiNGO – McCormick-based Algorithm for mixed-integer Nonlinear Global Optimization

Dominik Bongartz, Jaromił Najman, Susanne Sass, Alexander Mitsos*

Process Systems Engineering (AVT.SVT), RWTH Aachen University,
Forckenbeckstraße 51, 52074 Aachen, Germany

Abstract: MAiNGO is a deterministic global optimization software for solving mixed-integer nonlinear programs (MINLP). It is applicable to a wide range of MINLPs and has been shown to have computational advantages for classes of problems that admit reduced-space formulations. Furthermore, it can also serve as a framework for simulation and local optimization. Main algorithmic features of MAiNGO are the operation in the original variable space through the use of McCormick relaxations (i.e., no introduction of auxiliary variables) through MC++ (Chachuat et al., *IFAC-PapersOnline* 48 (2015), 981), custom relaxations for various functions (including several functions relevant to process systems engineering), and significant flexibility in model formulation. In addition to a basic branch-and-bound with some state-of-the-art bound tightening techniques like duality-based bound tightening and optimization-based bound tightening, it implements specialized heuristics for tightening McCormick relaxations as well as a multistart heuristic. This report summarizes the capabilities, algorithm, and software structure of the current version of MAiNGO (v0.1.12).

1 Introduction

Any nonlinear program (NLP) or mixed-integer nonlinear program (MINLP) with nonconvex functions can exhibit multiple local solutions. Local optimization methods (cf., e.g., [3]), which are usually based on gradients, can converge to any locally optimal solution and can even fail to find a feasible point at all for poor choices of initial points. Heuristic methods like genetic algorithms or simulated annealing converge to the global solution with probability one only as runtime goes to infinity (cf., e.g., [20]).

Deterministic global methods, in contrast, do guarantee finite convergence to the global solution given non-zero but arbitrary tolerances for feasibility (δ) and optimality (ϵ) specified by the user [11, 20]. Available deterministic global solvers like BARON [39], ANTIGONE [23], COUENNE [1], LINDOGlobal [19], MINOTAUR [21] or SCIP [42] are based on spatial Branch-and-Bound (B&B) [8]. They differ in many algorithmic details related to management of the B&B tree (cf., e.g., [1, 11, 20]), heuristics for bound tightening (cf., e.g., [1, 32]) and for finding good feasible points, as well as the way they construct relaxations for obtaining the required lower bounds (assuming minimization) on the objective function value (cf., e.g., [20, 23, 38]). However, to our knowledge all of them require explicit access to the algebraic structure of all functions involved in order to supply these relaxations and many do not support several useful functions (e.g., max, min, or trigonometric functions) [41].

The M*cCormick-based* A*lgorithm for mixed-*i*nteger* N*onlinear* G*lobal* O*ptimization* (MAiNGO) is a deterministic global optimization algorithm based on spatial B&B which relies on McCormick relaxations [22] and recent extensions [25, 36, 40, 44] that are implemented in the MC++ library of Chachuat et al. [7]. It can handle functions the algebraic form of which is not visible to the optimizer but whose function values, derivatives, relaxations and its subgradients are available at every point of the domain, and supports a relatively comprehensive library of intrinsic functions. In the following, we discuss the problem class that can be solved with MAiNGO, outline some core aspects of the algorithm and describe the basic structure of the implementation.

*Corresponding author: Alexander Mitsos
E-mail: amitsos@alum.mit.edu

2 Capabilities

MAiNGO can solve MINLPs of the following form, guaranteeing a solution that is δ-feasible and ϵ-optimal (adopting the definitions of [20]) or proving that no δ-feasible point exists for

$$\begin{aligned} \min_{\mathbf{x},\mathbf{y}} \quad & f(\mathbf{x},\mathbf{y}) \\ \text{s.t.} \quad & \mathbf{h}(\mathbf{x},\mathbf{y}) = \mathbf{0} \\ & \mathbf{g}(\mathbf{x},\mathbf{y}) \leqslant \mathbf{0} \\ & \mathbf{x} \in X \subset \mathbb{R}^{n_x} \\ & \mathbf{y} \in Y = \{0,1\}^{n_y} \end{aligned} \tag{1}$$

where $\mathbb{R}$ denotes the set of closed bounded intervals of $\mathbb{R}$. An important point is that the core of MAiNGO does not require access to the algebraic form of f, $\mathbf{g}$, and $\mathbf{h}$. The only requirement is that MAiNGO needs to be able to query function values and gradients at any given point $(\mathbf{x}^\mathsf{T},\mathbf{y}^\mathsf{T})^\mathsf{T} \in X \times Y$ as well as convex and concave relaxations and their subgradients on any given *node* defined by vectors of lower and upper bounds, $\left[\left(\mathbf{x}^{\mathrm{L}\mathsf{T}},\mathbf{y}^{\mathrm{L}\mathsf{T}}\right)^\mathsf{T},\left(\mathbf{x}^{\mathrm{U}\mathsf{T}},\mathbf{y}^{\mathrm{U}\mathsf{T}}\right)^\mathsf{T}\right] = \left[x_1^{\mathrm{L}},x_1^{\mathrm{U}}\right] \times ... \times \left[x_{n_x}^{\mathrm{L}},x_{n_x}^{\mathrm{U}}\right] \times \left[y_1^{\mathrm{L}},y_1^{\mathrm{U}}\right] \times ... \times \left[y_{n_y}^{\mathrm{L}},y_{n_y}^{\mathrm{U}}\right] \subset X \times Y$. In this sense, the functions f, $\mathbf{g}$, and $\mathbf{h}$ act in a similar way as external functions in modeling environments like GAMS [9]. The functions f, $\mathbf{g}$, and $\mathbf{h}$ could in fact also contain parts that are truly external in the sense that they have been implemented (and compiled) by a user who has made sure to supply gradients and relaxations - no matter how these are obtained. Note that Scott et al. [36] have shown that such relaxations can be integrated seamlessly with (generalized) McCormick relaxations.

The standard way for obtaining relaxations in MAiNGO is through the MC++ library [7] via the automatic propagation of McCormick relaxations through computer codes [22, 25, 36, 40]. To enable this propagation, f, $\mathbf{g}$, and $\mathbf{h}$ need to be factorable functions, i.e., they consist of a finite recursive composition of binary addition, multiplication, and intrinsic functions from a given library. Note that the library allows for multivariate outer functions as introduced in [40]. This library currently includes the following classes of functions:

- Univariate functions commonly included in modeling languages such as exponential, logarithm, power etc.,
- Multivariate functions including the binary product and division as well as the maximum and minimum of two variables [27, 40],
- Trigonometric and hyperbolic functions [35],
- Several functions that are commonly used in chemical process models [26], e.g., the logarithmic mean temperature difference (LMTD) and its reciprocal [24, 29].

While some of these could also be written as factorable functions themselves, including them as intrinsic functions allows the use of custom relaxations that can be substantially tighter than those obtained from propagating McCormick relaxations through the factorable representation.

For propagating McCormick relaxations, the main limitation on the computer codes implementing functions f, $\mathbf{g}$, and $\mathbf{h}$ is hence that they may not contain general conditional statements or iterative procedures with a priori unknown number of iterations. The former could, however, be circumvented through the use of the step functions introduced by [44]. The latter currently needs to be avoided through suitable model formulation (cf. [5]), since existing methods for the relaxations of implicit functions [37, 45] are currently not included.

The ability to supply procedural definitions of the functions in problem (1) through computer codes has two important benefits. First, the increased flexibility in model formulation can improve computational performance in global optimization since it enables convenient handling of reduced-space optimization formulations [5, 6, 12, 25, 35]. Second, the ability to use complex models implemented in a powerful programming language like C++ with the associated benefits compared to purely equation-oriented environments like GAMS makes MAiNGO a convenient

modeling framework beyond global optimization. Through the local solvers it interfaces (cf. Section 4), it can be used for simulating algebraic models (i.e., solving fully specified models without objective), or for local or heuristic global optimization using different local solvers that can be switched easily, as well as heuristics for generating initial points.

3 Algorithm

MAiNGO implements a spatial Branch-and-Bound (B&B) algorithm [8] with different heuristics for node selection (*best-first* by default), branching variable selection (by default taken as the variable with the largest diameter in the current node relative to the root node) and branching point selection (bisection by default). Detailed discussions of the convergence guarantees as well as a discussion of node selection and branching schemes can be found in [1, 11, 20, 38].

At each node k of the B&B tree, convex and concave relaxations and subgradients of the objective function and the constraint residuals are obtained through the propagation of McCormick relaxations [22] and their subgradients [25] as well as several extensions [27, 36, 40, 44] implemented in MC++ [7]. Other recent extensions like the differentiable modification of McCormick relaxations [15], relaxation of implicit functions [37] or reverse propagation [45] are currently not included. A discussion of the convergence properties of McCormick relaxations can be found in [4, 28, 30]. Optionally, the relaxations can be tightened through the heuristic introduced in [31], which uses the subgradients of the McCormick relaxations to tighten interval bounds at each factor. This, in turn, improves the relaxations of the following factors.

The relaxations are evaluated at one point (typically the midpoint) or multiple points (based on different heuristics). The lower bounding problem (LBP) is then constructed as described in [5] as a linearization of the McCormick relaxations at these evaluation points. The resulting linear program (LP^k) is solved to obtain a valid lower bound (LBD^k) on the optimal objective within the current node. Since no auxiliary variables are used, the LBP is formulated and solved in the same space as the original problem (1), which may have significantly lower dimensionality than that of the LBP resulting from the auxiliary variable method. At the same time, the resulting relaxations are equally tight as those generated by the auxiliary variable method [40], except in case the latter recognizes reoccurring subexpressions. Note, however, that MC++ [7] provides some functionality to also recognize such reoccurring subexpressions when generating McCormick relaxations.

As upper bounding problem (UBP), the original problem restricted to the current node is either solved locally (with gradients obtained via automatic differentiation) or the solution point of the LBP is simply checked for feasibility with respect to the original constraints. The number of iterations taken for upper bounding in each B&B node is limited to a user defined value.

Optionally, MAiNGO uses duality-based bound tightening (DBBT) and probing based on the multipliers of the LBP [33, 34], as well as optimization-based bound tightening (OBBT) [10] at every B&B node. The latter is also referred to as *standard range reduction* by [20] and is similar to the *contract* step in the algorithm of [46]. It consists of successively maximizing and minimizing each variable subject to the relaxed (and linearized) constraints, possibly including an objective function cut. In MAiNGO, the greedy heuristic and trivial filtering, both introduced by [10], are used for sorting OBBT runs and limiting the number of runs conducted. For filtering, the required minimum achievable improvement relative to the node width can be adjusted.

Before entering the B&B, MAiNGO conducts a pre-processing step. This step contains a check for special problem structures using the structure and dependency detection features of MC++ [7]. This way, linear programs (LP), mixed-integer linear programs (MIP), quadratic programs (QP), and mixed-integer quadratic programs (MIQP) can be recognized and handled appropriately (e.g., handed directly to CPLEX). Only in case none of these special problem types is recognized, the problem is treated as a general MINLP (or, possibly, NLP) and passed to the described solution procedure of MAiNGO. In this case, pre-processing optionally includes multiple rounds of OBBT as well as multiple local searches from randomly generated initial points.

Currently, only a very basic integer treatment is implemented in MAiNGO, which relies on simple branching of binary variables. Implementation and development of (mixed-)integer heuristics is part of future work for MAiNGO.

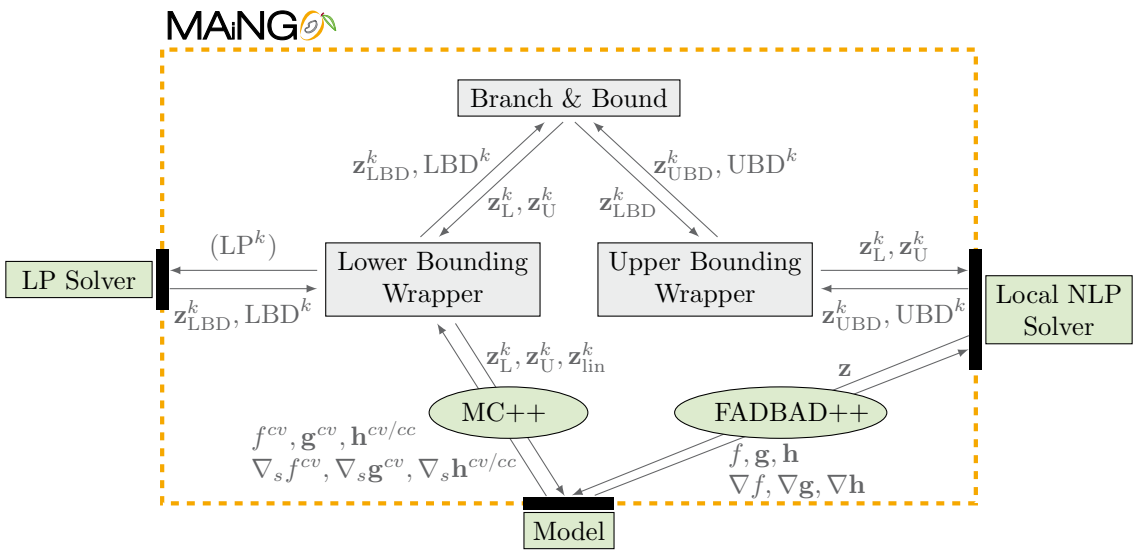

Fig. 1: Basic structure of MAiNGO. It interfaces *LP Solvers* and *Local NLP Solvers* for solving the lower and upper bounding problems, respectively. The algebraic structure of the *Model* is not visible to the core of MAiNGO and it hence acts as a black box in the sense that MAiNGO can only query function values and gradients (supplied by *FADBAD++* [2]) for the functions in problem (1) at a given point $\mathbf{z} = (\mathbf{x}^\intercal, \mathbf{y}^\intercal)^\intercal$ or convex and concave relaxations and the corresponding subgradients (supplied by *MC++* [7]) at a given linearization point $\mathbf{z}^k_{\mathrm{lin}}$ in the node $[\mathbf{z}^k_{\mathrm{L}}, \mathbf{z}^k_{\mathrm{U}}]$.

4 Structure

The structure of the solver is summarized in Figure 1. Denote by $\mathbf{z} = (\mathbf{x}^\intercal, \mathbf{y}^\intercal)^\intercal$ the vector of optimization variables of problem (1). To define an optimization problem, the user needs to supply an implementation of the *Model* containing the functions f, $\mathbf{g}$, $\mathbf{h}$ in the optimization problem (1), as well as the bounds $X \times Y$ on the optimization variables. The functions are currently implemented directly in `C++` using the data types provided by MC++ [7] for constructing a directed acyclic graph (DAG), but work is underway to enable text-based input as well. The only way MAiNGO interacts with the functions is to evaluate them with suitable data types to obtain either function values, gradients, or convex and concave relaxations and their subgradients.

After pre-processing, MAiNGO calls the *Branch-and-Bound* module which handles the B&B tree, including node selection, branching, and fathoming. It delegates the required LBP (defined by the variable bounds $[\mathbf{z}^k_{\mathrm{L}}, \mathbf{z}^k_{\mathrm{U}}]$ in node k) und UBP (defined by the variable bounds and the LBP solution point) to the *Lower Bounding Wrapper* and *Upper Bounding Wrapper*, respectively, which return the corresponding bounds on the objective function and solution points ($\mathbf{z}^k_{\mathrm{LBD}}$ and $\mathbf{z}^k_{\mathrm{UBD}}$).

The *Lower Bounding Wrapper* evaluates the functions encoded in the DAG using the data types of the `MC++` library [7] (with underlying interval extensions through FILIB++ [18]). Given lower and upper bounds $[\mathbf{z}^k_{\mathrm{L}}, \mathbf{z}^k_{\mathrm{U}}]$ on the optimization variables and a linearization point $\mathbf{z}^k_{\mathrm{lin}}$, convex and concave relaxations and the corresponding subgradients are thus queried for the functions in problem (1). Based on these linearizations, the relaxed problem (LP^k) is constructed and handed to an external *LP Solver*, currently CPLEX v12.8 [13] (other solvers are currently being added). The same applies to the problems arising in OBBT or probing (cf. Section 3).

The *Upper Bounding Wrapper* interfaces an external *Local NLP Solver*, choices for which currently include the SLSQP algorithm [16, 17] implemented in the NLOPT toolbox v2.5.0 [14] as well as IPOPT v3.12.0 [43]. The modular implementation allows for addition of other solvers in the future. The functions in the DAG are evaluated with the data types for automatic differentiation defined in the `FADBAD++` library [2] to obtain function values and gradients of f, $\mathbf{g}$, and $\mathbf{h}$ at a given point $\mathbf{z}$. The Upper Bounding Wrapper is also responsible for checking feasibility of the solution point $\mathbf{z}^k_{\mathrm{UBD}}$ returned by the local NLP solver (irrespective of the solver's return status). Finally, it also contains the heuristics for generating initial points for the local searches in pre-processing.

Acknowledgements We would like to thank Benoit Chachuat for providing MC++ and supporting us in extending it. Furthermore, we would like to thank Nikolaos V. Sahinidis for helpful discussions and sharing ideas. We would also like to thank our colleagues and numerous Bachelor's and Master's students for testing, feedback, and providing interesting case studies. Last but not least we would like to thank Hatim Djelassi for providing the idea for the MAiNGO logo. This work has received funding from the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) "Improved McCormick Relaxations for the efficient Global Optimization in the Space of Degrees of Freedom" MA 1188/34-1. The authors gratefully acknowledge additional funding by the German Federal Ministry of Education and Research (BMBF) within the "Kopernikus Project P2X: Flexible use of renewable resources – exploration, validation and implementation of 'Power-to-X' concepts".

References

[1] P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wächter. Branching and bounds tightening techniques for non-convex MINLP. *Optimization Methods and Software*, 24(4-5):597–634, 2009.

[2] C. Bendtsen and O. Stauning. *FADBAD++, a flexible C++ package for automatic differentiation. Version 2.1*, 2012. http://www.fadbad.com (Last accessed 07/12/2018).

[3] D. Bertsekas. *Nonlinear programming*. Athena scientific Belmont, 1999.

[4] A. Bompadre and A. Mitsos. Convergence rate of McCormick relaxations. *Journal of Global Optimization*, 52(1):1–28, 2012.

[5] D. Bongartz and A. Mitsos. Deterministic global optimization of process flowsheets in a reduced space using McCormick relaxations. *Journal of Global Optimization*, 69(4):761–796, 2017.

[6] D. Bongartz and A. Mitsos. Infeasible path global flowsheet optimization using McCormick relaxations. In A. Espua, M. Graells, and L. Puigjaner, editors, *Computer Aided Chemical Engineering: 27th European Symposium on Computer Aided Chemical Engineering (ESCAPE 27)*, pages 631–636, 2017.

[7] B. Chachuat, B. Houska, R. Paulen, N. Perić, J. Rajyaguru, and M. Villanueva. Set-theoretic approaches in analysis, estimation and control of nonlinear systems. *IFAC-PapersOnLine*, 48(8):981–995, 2015. https://omega-icl.github.io/mcpp/ (Last accessed 02/26/2018).

[8] J. Falk and R. Soland. An algorithm for separable nonconvex programming problems. *Management science*, 15:550–569, 1969.

[9] GAMS Development Corporation. *General Algebraic Modeling System (GAMS) Release 24.9.2*. Washington, DC, 2017.

[10] A. Gleixner, T. Berthold, B. Müller, and S. Weltge. Three enhancements for optimization-based bound tightening. *Journal of Global Optimization*, 67:731–757, 2017.

[11] R. Horst and H. Tuy. *Global optimization: Deterministic approaches*. Springer Science & Business Media, 2013.

[12] W. Huster, D. Bongartz, and A. Mitsos. Deterministic global optimization of the design of a geothermal organic rankine cycle. *Energy Procedia*, 129:50–557, 2017.

[13] International Business Machines Corporation. *IBM ILOG CPLEX v12.8*. Armonk, NY, 2017.

[14] S. Johnson. The NLopt nonlinear-optimization package. http://ab-initio.mit.edu/nlopt (Last accessed 09/13/2018).

[15] K. Khan, H. Watson, and P. Barton. Differentiable McCormick relaxations. *Journal of Global Optimization*, 67(4):687–729, 2017.

[16] D. Kraft. A software package for sequential quadratic programming. Technical Report DFVLR-FB 88-28, Institut für Dynamik der Flugsysteme, Oberpfaffenhofen, 1988.

[17] D. Kraft. Algorithm 733: TOMP–Fortran modules for optimal control calculations. *ACM Transactions on Mathematical Software*, 20(3):262–281, 1994.

[18] M. Lerch, G. Tischler, J. Wolff von Gudenberg, W. Hofschuster, and W. Krämer. FILIB++, a fast interval library supporting containment computations. *ACM Transactions on Mathematical Software (TOMS)*, 32:299–324, 2006. http://www2.math.uni-wuppertal.de/~xsc/software/filib.html (Last accessed 07/16/2018).

[19] Y. Lin and L. Schrage. The global solver in the LINDO API. *Optimization Methods & Software*, 24(4-5):657–668, 2009.

[20] M. Locatelli and F. Schoen. *Global optimization: theory, algorithms, and applications*, volume 15. SIAM, 2013.

[21] A. Mahajan, S. Leyffer, J. Linderoth, J. Luedtke, and T. Munson. Minotaur: A mixed-integer nonlinear optimization toolkit. Technical report, Mathematics and Computer Science Division, Argonne National Laboratory, 2017. http://www.optimization-online.org/DB_HTML/2017/10/6275.html (Last accessed 07/17/2018).

[22] G. McCormick. Computability of global solutions to factorable nonconvex programs: Part I – Convex underestimating problems. *Mathematical Programming*, 10:147–175, 1976.

[23] R. Misener and C. Floudas. ANTIGONE: Algorithms for coNTinuous / Integer Global Optimization of Nonlinear Equations. *Journal of Global Optimization*, 59:503–526, 2014.

[24] M. Mistry and R. Misener. Optimising heat exchanger network synthesis using convexity properties of the logarithmic mean temperature difference. *Computers & Chemical Engineering*, 94:1–17, 2016.

[25] A. Mitsos, B. Chachuat, and P. Barton. McCormick-based relaxations of algorithms. *SIAM Journal on Optimization*, 20(2):573–601, 2009.

[26] J. Najman, D. Bongartz, and A. Mitsos. Convex and concave envelopes of functions of interest in chemical engineering. 2018. In preparation.

[27] J. Najman, D. Bongartz, A. Tsoukalas, and A. Mitsos. Erratum to: Multivariate McCormick relaxations. *Journal of Global Optimization*, 68(1):219–225, 2017.

[28] J. Najman and A. Mitsos. Convergence analysis of multivariate McCormick relaxations. *Journal of Global Optimization*, 66:597–628, 2016.

[29] J. Najman and A. Mitsos. Convergence order of McCormick relaxations of LMTD function in heat exchanger networks. In *26th European Symposium on Computer Aided Process Engineering*, pages 1605–1610, 2016.

[30] J. Najman and A. Mitsos. On tightness and anchoring of McCormick and other relaxations. *Journal of Global Optimization*, 2017. In press. DOI: 10.1007/s10898-017-0598-6.

[31] J. Najman and A. Mitsos. Tighter McCormick relaxations through subgradient propagation. 2018. Submitted for publication. arXiv:1710.09188.

[32] Y. Puranik and N. Sahinidis. Domain reduction techniques for global NLP and MINLP optimization. *Constraints*, 22(3):338–376, 2017.

[33] H. Ryoo and N. Sahinidis. Global optimization of nonconvex NLPs and MINLPs with applications in process design. *Computers & Chemical Engineering*, 19(5):551–566, 1995.

[34] H. Ryoo and N. Sahinidis. A branch-and-reduce approach to global optimization. *Journal of Global Optimization*, 8(2):107–138, 1996.

[35] A. Schweidtmann and A. Mitsos. Global deterministic optimization with artificial neural networks embedded. 2018. Submitted for publication. arXiv:1801.07114.

[36] J. Scott, M. Stuber, and P. Barton. Generalized McCormick relaxations. *Journal of Global Optimization*, 51:569–606, 2011.

[37] M. Stuber, J. Scott, and P. Barton. Convex and concave relaxations of implicit functions. *Optimization Methods & Software*, 30:424–460, 2015.

[38] M. Tawarmalani and N. Sahinidis. *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming*. Kluwer Academic Publishers, Dordrecht, Netherlands, 2002.

[39] M. Tawarmalani and N. Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, 103(2):225–249, 2005.

[40] A. Tsoukalas and A. Mitsos. Multivariate McCormick relaxations. *Journal of Global Optimization*, 59:633–662, 2014.

[41] S. Vigerske. *MINLP Solver Technology*, 2017. https://www.gams.com/~stefan/2017_minlp.pdf (Last accessed 07/06/2018).

[42] S. Vigerske and A. Gleixner. SCIP: Global optimization of mixed-integer nonlinear programs in a branch-and-cut framework. *Optimization Methods and Software*, 33(3):563–593, 2018.

[43] A. Wächter and L. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[44] A. Wechsung and P. Barton. Global optimization of bounded factorable functions with discontinuities. *Journal of Global Optimization*, 58(1):1–30, 2014.

[45] A. Wechsung, J. Scott, H. Watson, and P. Barton. Reverse propagation of mccormick relaxations. *Journal of Global Optimization*, 63(1):1–36, 2015.

[46] J. Zamora and I. Grossmann. Continuous global optimization of structured process systems models. *Computers & Chemical Engineering*, 22(12):1749–1770, 1998.